Real-time implementation of a blind authentication method using self-synchronous speech watermarking

Parera, Oriol. Megias, David.

{Oriol.PareraF, DMegias}@uoc.edu

Abstract

A blind speech watermarking scheme that meets hard real-time deadlines is presented and implemented. In addition, one of the key issues in these block-oriented watermarking techniques is to preserve the synchronization. Namely, to recover the exact position of each block in the mark extract process. In fact, the presented scheme can be split up into two distinguished parts, the synchronization and the information mark methods. The former is embedded into the time domain and it is fast enough to be run meeting real-time requirements. The latter contains the authentication information and it is embedded into the wavelet domain. The synchronization and information mark techniques are both tunable in order to allow a configurable method. Thus, capacity, transparency and robustness can be configured depending on the needs. It makes the scheme useful for professional applications, such telephony authentication or even sending information throw radio applications.

Keywords: real time, speech watermarking, authentication, wavelet, Daubechies

1 Introduction

Nowadays authentication is an important concern for communication companies. Furthermore, it is a demand task because the hidden message must be unfindable, robust to compression algorithms such the G.711, and go unnoticed into a speech signal.

Watermarking has been researched as potential solution to multiple and scattered issues in speech[1, 4, 5, 6, 8].

Generally, watermarking in frequency domain produce robust watermarks. Nevertheless two possible problems arise, desynchronization and high computational cost. Desynchronization is because the secret message is clipped in time-domain and therefore is difficult to recognize the clipped part in wavelet. On the other hand, even nowadays there are algorithms like the Fast Fourier transform, there is not lower complexity than the native domain, e.g. time-domain. However, using these schemes is difficult to meet real-rime requirements.

Specifically, solutions proposed in [4] takes advantage of non-voiced speech (silences). Its capacity is placed around 450bits/s and the information is hidden into the time-domain. Despite these characteristics so good, the detection of the synchronization mark takes at least 0,2 seconds. The method proposed in [5] takes advantage of a spread spectrum and it is robust to multiple attacks, such the compression. Furthermore, it depicts a high capacity (around 706kbps). In this case, the embedding method alters the speech signal by combining frequency values with the original speech followed by accumulating them. Namely, it seems difficult to implement meeting real-time deadlines.

Methods detailed in [1, 8] use the wavelet domain to embed the information data. The former takes advantage of a logarithmic quantization while the latter uses a multiplicative method. The proposed scheme uses the logarithmic quantization from [1] because that space is more imperceptible to the human ear. Furthermore, in order to meet real-time requirements the proposed scheme complements the wavelet watermarking proposed in [1] with a fast time-domain synchronization detailed below.

Proposal for self-synchronized watermarking methods have been described in [2, 7, 9, 10]. The method detailed in [10] synchronizes after computing the DFT. Namely, if the proposed scheme does not hide the information mark using the same characteristics -such position and length- the time to execute the DFT per each received sample would not work meeting real-time requirements. In the case of [9] shows a synchronization method prepared for images attacks. Despite this is a powerful method, it has been thought for images and it uses 2-d characteristics that a speech signal does note have.

On the other hand, the methods [2, 7]. The former is a proposal for blind audio watermarking based on time-domain and FFT Amplitude. It uses a very interesting method for synchronization prepared for meeting real-time requirements and in fact, the proposed scheme uses this synchronization method for this reason. The latter is a proposal for hiding secret messages in audio robust to mp3, low-pass filtering, and clipping. However, the high computational cost problem arises again in this technique.

Imperceptibility, robustness, capacity, security and real time are five requirements to meet by speech watermarking systems. Namely, the system must be imperceptible to human senses (imperceptible), and able to exchange speech information without losing the hidden data (robustness), and must contain data enough for an specific application (capacity). The information must be hidden and impossible to get by third parties (security), and must meet real-time requirements because a Speech (as a telephonic call) is a real-time media.

A tradeoff between capacity, imperceptibility and robustness against intentional or unintentional channel attacks is described in [3]. Hence, it is mandatory a decision to prioritize these characteristics. In the proposed scheme it is considered preferable to lose some of the embedded marks as far as no perceptible distortion is detected in the marked speech. However, the system must be robust enough such that many of the embedded marks are extracted after performing typical processing signal attacks, as the compression G.711.

The objective of the watermarking system presented in this paper is to introduce a method self-synchronized in the time-domain and using configurable embedding method into the wavelet domain implemented on a system meeting real-time requirements and robust to compression G.711.

In fact, the proposed scheme is a combination between two methods from the above literature [1, 2], detailed below.

- The self-synchronized method in [2], that provides a robust and transparent synchronization that even meets real-time deadlines.
- A minor variation of the wavelet method detailed in [1], because this method deals with high robustness and transparency. Although this method is difficult to compute meeting real-time requirements, the proposed implementation (see the Appendix) manages to work it out by adding an imperceptible delay.

Thus, the proposed implementation adds a constant and imperceptible delay to the transmission. This delay is added by the emitter embedded into reception of the voice and sending it to the receptor. Furthermore, this delay is a parameter of the method and it is decided in function of the number of secret bits to hide. In fact, the mentioned delay corresponds to the time spent by the emitter in order to embed the secret bits.

2 Proposed method

The proposed method contains two important parts, i.e., the embedding and the extracting processes.

2.1 Embedding process

2.1.1 Synchronization embedding method

The synchronization marks are a very relevant issue into the speech watermarking schemes and they are used for identify where the information mark starts. Therefore, if the synchronization mark is not detected in the extracting process the information mark will never be extracted.

If the embedding and detection algorithms work with longer blocks of data, the efficiency of the system in both memory and CPU time increases significantly. However, the synchronization marks must be easy to detect if an efficient watermark scheme is required. Obviously, this is mandatory into a system design that meets real-time requirements.

The synchronization marks proposed in [2] are easy to compute, both to hide and to extract methods. Thus, the implemented synchronization method is exactly the method proposed in [2]. The algorithm, detailed below, is executed for embedding each single bit.

1. Take one frame of a given length from the speech signal where the default length is four samples.

2. Compute the average of the inner samples and the external samples of the frame (λ_i and λ_e) as follows.

   (a) $\lambda_i = \frac{1}{n_{syn}-2} \sum_{j=p+1}^{p+n_{syn}-2} s_j$

   (b) $\lambda_e = \frac{1}{n_{syn}-2} \sum_{j=p+1}^{p+n_{syn}-2} s_j$

3. Calculate the minimum expected difference between the average of the inner samples and the external samples, expressed as δ.

   (a) $\delta = max\left\{\delta_{min}, \phi|S_{ext}|\right\} \forall\delta_{min} > 0, \phi > 0$

4. Embed the secret bit as follows.

   (a) Embed a 0:

      i. When "λ_e< λ_i + δ" then the distance (d) is calculated as "d = λ_i + δ - λ_e" followed by subtracting this distance from each inner sample of the frame.

      ii. Otherwise the samples of this frame are not modified.

   (b) Embed a 1:

      i. When "λ_i < λ_e + δ" then the distance (d) is calculated as "d = λ_e + δ - λ_i" followed by adding this distance to each inner sample of the frame.

      ii. Otherwise the samples of this frame are not modified.

5. Replace the original frame by the altered one into the original speech signal

Note that this scheme assures that the difference between the average of the inner and the external samples is at least δ for each synchronization bit. The Figure 1 depicts this fact.

2.1.2 Information mark embedding method

The watermark is embedded into the DWT domain following the manner explained in[1], taking advantage of the logarithmic quantization. In fact, the method in [1] computes the DWT using the Daubechies 10-Tap coefficients and this is changed in this implementation, which is configurable for computing the DWT using the coefficients from Daubechies DB01 to Daubechies DB20.

In order to embed secret information into the speech, it is divided into frames and a group of bits are embedded into each corresponding frame. Following the method proposed in [1], each wavelet sample is mapped into the logarithm domain and altered depending on the secret bit.

The embedding steps are described below.

1. Take one frame of a given length from the speech signal.

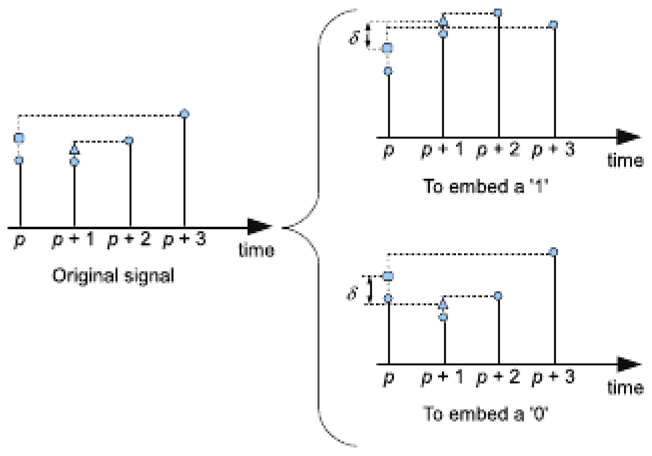

Fig. 1: Synchronization bit hidden into 4 samples, from [2]

2. Compute the first level Daubechies wavelet transform of the frame -dbxx where xx vary from 00 to 20, depending on the software configuration.
3. Divide the cA samples into groups of a given length.
4. Hide a secret bit into each block, altering each sample of the block as follows.
   (a) Hide a 0: m_i= sign(c_i)10^($\delta \lfloor(\log_{10}|c_i|)/\delta\rfloor$)
   (b) Hide a 1: m_i= sign(c_i)10^($\delta \lfloor(\log_{10}|c_i|)/\delta\rfloor + \delta/2$)
5. Apply the inverse DWT.
6. Replace the original frame by the marked one into the original speech signal.

m_i is the marked value where c_i represents a cA sample. cA samples represents the output of the low-pass filter of the wavelet transform. Note that the output of the high-pass filter of the wavelet transform is not altered. δ is the quantized value.

The wavelet transform coefficients happen to have a small dynamic range for some speech sequences. As such, upon applying the logarithm mapping and fixing process, they are converted to just zero and one. To prevent this, the implemented method multiplies the wavelet samples by a scale factor and then after the embedding process, scales them down by the same factor. By default, the implemented application uses a scale factor 1000.

2.2 Extracting process

2.2.1 Synchronization extracting method

While the marked speech signal is received, the process for detecting the synchronization mark is constantly executed. Therefore, the execution of this method must be fast. The method proposed in [2] meets this requirement.

The algorithm for detecting the synchronization mark [2] into the marked speech is detailed below. This algorithm is executed for extracting each single bit.

1. Take one frame of a given length from the speech signal where the default length is four samples.
2. Compute the average of the inner samples and the external samples of the frame (μ_i and μ_e) as follows.
   (a) $\mu_i = \frac{1}{n_{syn}-2} \sum_{j=p+1}^{p+n_{syn}-2} t_j$
   (b) $\mu_e = \frac{1}{n_{syn}-2} \sum_{j=p+1}^{p+n_{syn}-2} t_j$
3. Extract the secret bit.
   (a) Extract a 0: When $\mu_i <= \mu_e$.
   (b) Extract a 1: Otherwise.

2.2.2 Information mark extracting method

Once the synchronization mark has been extracted, the application extracts the watermark data hidden into a frame by computing the wavelet samples of the marked signal and then the hidden bit is extracted in function of the result got of calculating the difference between the marked samples and the rounded marked samples. The steps are listed below.

1. Take the marked frame from the marked speech.
2. Compute the first level Daubechies wavelet transform -depending on the software configuration- of the frame.
3. Divide the cA samples into groups of a given length.
4. Extract the embedded bit in each marked wavelet sample as follows.
   (a) Extract a 0, when $||\delta \log_{10}|m_i|| - \mathrm{round}(\delta \log_{10}|m_i|)| < 0.25$
   (b) Extract a 1, otherwise.
5. Calculate the real extracted bit per each group by summarizing all the extracted zeros and all the extracted ones followed by comparing those values and assuming that the real bit per each group is the greater value between both summarizes.

m_i is the marked value where c_i represents a cA sample. cA samples represents the output of the low-pass filter of the wavelet transform. Note that the output of the high-pass filter of the wavelet transform is not altered. δ is the quantized value.

3 Experimental results and Discussion

In order to evaluate the performance of the proposed technique in real conditions, it has been implemented for an embedded system based on a DSP architecture meeting hard real-time deadlines. The implemented real-time application details are located in the appendix.

Experimental executions are performed on different types of speech including male and female voices talking Spanish and even English. To illustrate the results, fifteen audio files in wav format are used. Furthermore, these audio files are mono, sampled at 4KHz and quantized with 16 bits. Moreover, the speakers output pin from the computer is connected to the microphone input pin of the DSP in order to process these speech signals meeting real-time requirements.

The suggested method has different properties to take into account and, among them, capacity, transparency, robustness, and real-time requirements are the most important ones. In this section we discuss those properties of the proposed speech watermarking scheme. Thus, before presenting these results, the following section provide a brief summary about the tuning configuration applicable to the suggested technique. Note that all of these parameters can be tuned in the implemented real-time application on the DSP.

3.1 Experimental settings

The tuning parameters of the suggested method chosen in this paper are detailed as follows.

3.1.1 Synchronization tuning parameters

The synchronization parameters are used to configure the behavior of the proposed synchronization method. The tables 1 and 2 depict this behavior. However, this configuration briefing is detailed below.

1. The "synchronization code" and its length. Specifically, each and every one of the synchronization code bits are hidden/extracted in order to determine the position where the wavelet-domain watermark with the hidden secret bits begins. Furthermore, the synchronization code length vary in this experimental tests set in the range of 8, 16, 24 and 32 hidden bits in the time-domain.

2. The alteration intensity (ϕ) applied in the proposed synchronization method. Depending on this setting, the marked speech corresponding to the synchronization bits will be strongly altered. Hence, high values for this

parameter could be perceptible or even destruct the original speech signal and, by the other hand, low values could become the mark unextractable.

3. The number of speech samples used to hide each single secret bit. In order to execute the proposed technique into a system meeting real-time requirements and, taking into account that the extracting process of the synchronization marks requires high computational costs, the proposed experimental tests use four samples to hide each single synchronization bit.

The tables 1 and 2 depicts the synchronization mark behavior in function of these parameters. The former table shows the number of false positive found into a speech that no marks have been hidden. Specifically, short synchronization marks (8 and 16 bits) generate false positive. It happens because even without altering the original speech signal the synchronization message is already coded into the samples. However, it does not happen for longer synchronization marks. The latter table shows the number of successfully extracted marks varying in function of the intensity constant (ϕ). In particular, when ϕ is lower than 0.5 the synchronization mark is not successfully extracted. However, it happens because the marked signal was not altered with the needed intensity and then the extracting process does not work for some bits.

Hence, the optimal parameters for the synchronization method that will be used for the experimental results will be ϕ= 0,5 and synchronization length of 32 bits.

PHI Constant	Bits of Synchronization	False positives found	Relative false positives found
0.25	8	425	0.51%
0.25	16	5	0.05%
0.25	24	0	0
0.25	32	0	0
0.5	8	425	0.51%
0.5	16	5	0.05%
0.5	24	0	0
0.5	**32**	**0**	**0**

Tab. 1: False positives found into Speeches of 10 seconds

3.1.2 Information mark tuning parameters

The watermark proposed method parameters are used to configure the behavior of the watermark technique. However, this configuration briefing is detailed below.

1. The authentication embedded message length. In any case, the proposed experimental results assumes that four bytes (32 bits) are enough to authenticate any potential user of the system. In addition, this system could manage around 4294,97 million users.

PHI Constant	Bits of Synchronization	Successfully extracted bits	Successfully extracted marks
0.25	8	99.91%	83%
0.25	16	99.91%	67%
0.25	24	99.91%	53%
0.25	32	99.91%	45%
0.5	8	100%	100%
0.5	16	100%	100%
0.5	24	100%	100%
0.5	**32**	**100%**	**100%**

Tab. 2: Successfully bits and marks extracted from Speeches of 10 seconds

2. The frame size. The frame size parameter refers to the number of speech samples that will be used to compute the proposed information mark embedding method. Note that a tradeoff between the number of hidden bits and computational costs takes place in deciding the frame size. In the proposed experiments the tested frames are in the range of 64, 128 or 256 samples.

3. The wavelet transform computed to each single frame. The DWT transform applied in the real-time implantation of the proposed method can be tuned for using the Daubechies dbxx coefficients where dbxx can be from db01 to db20. Although the implemented application can is prepared for working with all of those DWT Daubechies coefficients, in this experimental results the db10 and db15 have been used.

4. The scale factor applied to the logarithmic quantization. The scale factor applied to the proposed watermarking technique is 1000.

5. The logarithmic quantization (δ) in the proposed watermarking method. Depending on this setting, marked speech corresponding to the watermarked bits will be strongly altered. Hence, high values for this parameter could be perceptible or even destruct the original speech signal and, by the other hand, low values could make the mark unextractable. Furthermore, this setting must be in accordance with the scale factor. However, for a scale factor about one thousand range of values for the logarithmic quantization should be between one and nine.

The table 3 depicts the watermark method behavior by varying these parameters.

3.2 real-time requirements

Meeting real-time requirements means that the speech signal must be continuously processed. In addition, the speech signal can contain a short delay (less than half second) without being annoying and keeping imperceptibility but it is important that the processing rate remains constant.

DBXX	δ	hidden bits	blocks of 64 s	blocks of 128 s	blocks of 256 s
DB10	1	1	100%	100%	100%
DB10	1	2	100%	100%	100%
DB10	1	4	100%	100%	100%
DB10	1	8	66.67%	100%	100%
DB10	1	16	13.33%	13.33%	100%
DB10	5	1	100%	100%	100%
DB10	5	2	100%	100%	100%
DB10	5	4	73.33%	100%	100%
DB10	5	8	26.67	53.33%	100%
DB10	5	16	0.00%	6.67%	33.33%
DB14	1	1	100%	100%	100%
DB14	1	2	100%	100%	100%
DB14	1	4	100%	100%	100%
DB14	1	8	66.67%	100%	100%
DB14	1	16	13.33%	13.33%	100%
DB14	5	1	100%	100%	100%
DB14	5	2	100%	100%	100%
DB14	5	4	73.33%	100%	100%
DB14	5	8	26.67%	53.33%	100%
DB14	5	16	0.00%	6.67%	33.33%

Tab. 3: Information mark bits successfully extracted

In order to meet these requirements, it is important to measure the spent time by the system processes. The table 4 depicts these time.

In fact, the real-time requirements have two different approaches depending on whether it is being executed the embedding process or the extracting process. These two approaches are detailed below.

3.2.1 Embedding process

The time spent on computing each single frame is higher than the time contained by that single frame of speech signal. For instance, computing the proposed and implemented watermark embedding method into each single frame of 256 samples takes 60 ms (see Table 4) while 256 samples represents 32,25 ms of speech signal. In order to achieve time enough to compute the embedding method, the implemented application is provided with a circular buffer where to keep frames. Consequently, the applications adds a delay which depends on the used frame size and the number of frames buffered, as it is shown below.

- $delay = fs * nb/8$

The delay is expressed in ms. fs is the frame size and, nb is the number of buffers. In addition, the number of buffers used for these tests is four. Therefore, buffering four frames constantly adds the delay shown in table 5.

Method	Blocks length	DWT	iDWT	Embedding	Extracting
DB10	64	07.74	05.01	16.20	11.20
DB10	128	12.14	07.78	25.40	17.65
DB10	256	17.72	14.15	40.70	25.80
DB14	64	07.53	03.58	14.17	10.24
DB14	128	18.90	10.34	37.30	24.52
DB14	256	28.22	18.84	60.00	40.15
SYN	8	-	-	less than 00.01	less than 00.01
SYN	16	-	-	less than 00.01	less than 00.01
SYN	24	-	-	less than 00.01	less than 00.01
SYN	32	-	-	less than 00.01	less than 00.01

Tab. 4: Times expressed in ms

Frame size (samples)	computational time (ms)	delay (ms)	max. frames per second
64	around 20	32	31
128	around 45	64	15
256	around 70	128	7

Tab. 5: Computational time vs added delay

3.2.2 Extracting process

The implemented extracting process is not meeting real-time requirements because it is assumed that the receiver can be saving the marked speech signal and processing it using computers with high computational capacities. Actually, it is intended to implement this functionality in a future work.

3.3 Capacity

Capacity is defined by the number of bits embedded in one second of the speech signal. Note that the required capacity depends on the number of bytes needed for authenticating and the maximum time spent to be authenticated. For instance, to authenticate a speech signal of a call in about one second the required capacity would be about 32 bps at least (assuming that 4 bytes are enough to authenticate any user of the system).

Following the above definition, the capacity of the proposed method is defined by the number of frames computed per second and the number of bits hidden per frame. Consequently, the system must be tuned to guarantee the required capacity per each case.

Thus, the maximum capacity depends directly on the maximum number of frames hidden per second and the number of bits hidden into each frame.

In addition, the relation between capacity (σ), the number of bits hidden (ξ) per each watermark and, the maximum number of frames per second (κ) is expressed below.

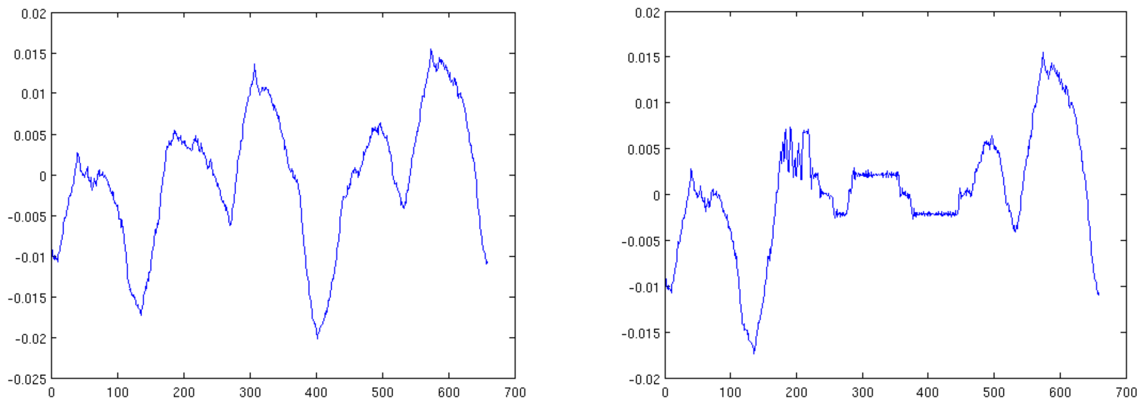

Fig. 2: Synchronization: Original Signal and Marked Signal

- $\sigma = \xi \left\lfloor \frac{1000}{delay} \right\rfloor = \xi \cdot \kappa$

For instance, the implemented method can hide seven frames per second when the frame size is 256 samples (see the Table 5) and then hiding 16 bits per each mark the capacity would be 112 bps. ($112 = 16 \cdot 7$)

3.4 Transparency

The proposed technique is similar to [1] and also their results should be similar. In fact, the PSEQ-MOS obtained using the method proposed in [1] is around 4. The scale of values is 1 (bad) and 5 (excellent). Unfortunately, he PSEQ-MOS could not be quantified and compared to [1] by lack of access to the software. Although the transparency could not be tested and quantified, by listening to the marked speech signal, it seems that the embedded information is human imperceptible. Furthermore, the transparency tests will be executed in the future.

However, the Figures 2 and 3 depict graphically two examples of distortion caused by coding (the synchronization and the information mark) using 32 bits for the synchronization and the Daubechies DB14 hiding 8 bits for the information mark.

3.5 Robustness

The scheduled attack to be tested into the experimental results for the proposed authentication technique is G.711 compression. Unfortunately, the robustness

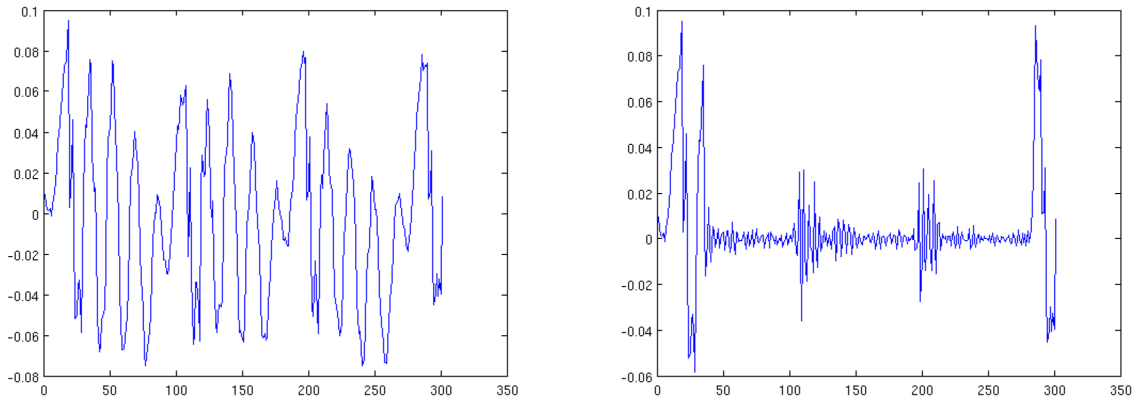

Fig. 3: Information mark: Original Signal and Marked Signal

could not be tested due to lack of time.

4 Limitations and next steps

Although listening to the marked speech it seems imperceptible, the transparency tests are not performed. These tests will be performed in future work. In addition, the robustness tests will be performed in future work by testing G.711 compression.

An extracting method meeting real-time requirements would be useful for extracting the mark using an embedded system. It would allow, for instance, a bidirectional communication.

The bitrate and the capacity could be increased by executing in parallel more than one embedding process. Although the CPU use is about 25% it would not work using the implemented application because the embedding process uses critical memory (in general, global variables) that should be protected by semaphores for executing more than one thread. By extension, even the extracting process could be improved using more than one thread for extracting the mark. Note that only one thread would test for finding the mark and this task would launch other threads for computing the extracting method.

5 Conclusions

This paper presents a real-time implementation of a self-synchronized speech watermarking scheme. The scheme combines a time-domain embedding of the synchronization marks with a wavelet domain embedding of the authentication mark. Furthermore, the time domain synchronization mark does not produce audible clicks.

In addition, the information marking method which computes the wavelet transform, is configurable for using different Daubechies coefficients. Therefore, the proposed implementation guarantees that any of the configurations will run successfully meeting hard real-time deadlines. In fact, this is guaranteed by adding a delay into the marked speech signal transmission that is used for computing the embedding method.

The proposed scheme is shown to provide excellent capacity results (around 112 bps) more than enough for authenticating methods. Unfortunately, the experimental results of transparency and robustness are not performed and therefore, there are no objective conclusions about them. However, they will be studied into a future work.

References

[1] Mehdi Fallahpour, David Megias, Hossein Najaf-Zadeh. Adaptative speech watermarking in wavelet domain based on Logarithm**.** Proceedings of the International Conference of Security and Cryptography (SECRYPT 2012).

[2] David Megias, Jordi Serra-Ruiz, Mehdi Fallahpour. Efficient self-synchronized blind audio watermarking system based on time domain and FFT amplitude modification**.**

[3] Jana Dittman, David Megias, Andreas Lang, Jordi Herrera-Joancomarti. Theoretical Framework for a Practical Evaluation and Comparison of Audio Watermarking Schemes in the Triangle of Robustness, Transparency and Capacity**.**

[4] Hofbauer, k. Kubin, G. Kleijn, W.B. speech watermarking for Analog Flat-Fading Bandpass Chanel. Audio, Speech, and Language processing, IEEE Transactions. Nov. 2009. Volume 17, Issue 8. Pages 1624 - 1637.

[5] Qiang Cheng, Jeffrey Sorensen. Spread spectrum signaling for speech watermarking**.** Acoustics, Speech, and Signal rocessing. 2001. Volume 3. Pages 1337 - 1340.

[6] Mohammad Ali Akhaee. Nima Khademi Kalantari. Farokh Marvasti. Robust audio and speech watermarking using Gaussian and Laplacian modeling**.** Signal processing 90 (2010). Pages 2487 - 2497.

[7] Martinez-Noriega, Raul. Nakano, Mariko. Yamaguchi, Kazuhiko. Self-Synchronous Time-Domain Audio Watermarking Based on Coded-Watermarks**.** Signal Processing (Oct. 2010). Pages 135 - 138.

[8] Nima Khademi Kalantari. Mohammad Ali Akhaee. Seyed Mohammad Ahadi. Hamidreza Amindavar. Robust Multiplicative Patchwork Method

for Audio Watermarking. ieee transactions on audio, speech, and language processing, vol. 17, no. 6. august 2009.

[9] ZHANG Li, QIAN Gong-bin, CHEN Li-min, JI Zhen, LI Xia. Self-synchronization Adaptive Blind Image Watermarking Technique with Characteristics of Original. Ma, Proc. of SPIE Vol. 6044.

[10] Ryuki Tachibana. Shuichi shimizu. Taiga Nakamura. Seiji Kobayashi. An audio watermarking method robust against time- and frequency - fluctuation.

Appendix

Here is presented an implementation for the proposed method that is written under the license GPL-2. Hence, it is free software and it can be executed, copied, modified and, redistributed. In fact, the wavelet transformation has been copied and modified from the Scilab Wavelet Toolbox (SWT), which is implemented in C and its license is already the GPL-2.

Furthermore, this application can run on two architectures. The former is a DSP architecture, particularly the DSP TMS3205416 architecture, meeting hard real-time requirements. Note that this configuration is thought as the main application. The latter runs on a x86 platform without real-time deadlines to meet by the scheduler. Note that this configuration has been used mainly for testing the proposed technique and it works reading/writing wav files.

In order to choose the architecture in compilation time, there is a preprocessor constant in the arq.h file. Equally, the tuning parameters detailed into the experimental settings section can be configured by modifying the parameters.h file.

The following sections depict the embedding and the extracting process design for being able to run meeting hard real-time deadlines.

Embedding process meeting hard real-time deadlines

The embedding process is composed by the tasks detailed below. Note that the frame size and the data given in this section is thought for frames of 256 samples.

In any case, the challenge to solve in order to meet hard real-time requirements although the calculating time is higher than the input/output bitrate, is that the application uses 4 buffers of 256 samples each one, disposed in circle, where the speech signal is loaded. Thus, there is a delay between the instant that the hardware gets a speech sample and the instat that puts that sample out.

The tasks that compose the embedding process are detailed below.

1. The main procedure, which executes the hardware and software initializations and prepares the hardware for the Input/Output.

2. A software interruption (SWI), called “reading”. The DSP manages the Input/Output launching a hardware interruption when a sample is received/has to be sent. In order to be detached from the application, these hardware interruptions interacts with an intermediate input/output buffers placed between these hardware interruptions and the application. Therefore, it appears the need of having a software interruption, that is called when the Input/Output buffer is full/empty (256 samples for instance). The execution of this software interruption uses a 5% of the CPU. Thus, the first execution of the SWI reads the 256 input samples placed in the input buffer and loads them into the application buffer0. The second execution into the application buffer1, the third execution into the application buffer2, until the last execution that loads the input data into the last application buffer. Then, the next input data is loaded again into the first application buffer, then the second application buffer, the third application buffer, and so... This way, the circular application buffer keeps always four speech frames loaded.

3. The task “synchronization”, that hides the synchronization mark into the speech. This task is attached to the buffer0 to hide there the synchronization mark. Therefore, although it is required to be hidden a synchronization mark, the task must wait for a semaphore until this buffer is ready with speech information. In addition, the software interrupt detailed above posts this semaphore when the buffer is ready with speech information to be processed. Note that this task is just an infinite loop that just waits for a semaphore and hides synchronization marks into the buffer0.

4. The task “encoder”, that embeds a watermark into a given frame placed in the buffer1. That is, the buffer1 is attached to this task. Extrapolating from the above task, this buffer is also a infinite loop that waits for a semaphore which the software interruption will post when it is need and, in that case, computes the proposed embedding watermark method.

5. The task ”sendToHost”, that sends to the host all the values that go the the speakers. It is necessary to send all the marked speech data to the host in order to get it back in order to execute the extraction process or just study the experimental results. Note that this task is used only for testing and in case that it was inserted into production this task should not be executed. Therefore, there is a preprocessor constant to enable/disable this task.

The figure 4 depicts these operations.

Extracting process meeting hard real-time deadlines

Although the extracting process is not implemented, it is thought how it would work. In fact, this process does not need to generate a delay because the input

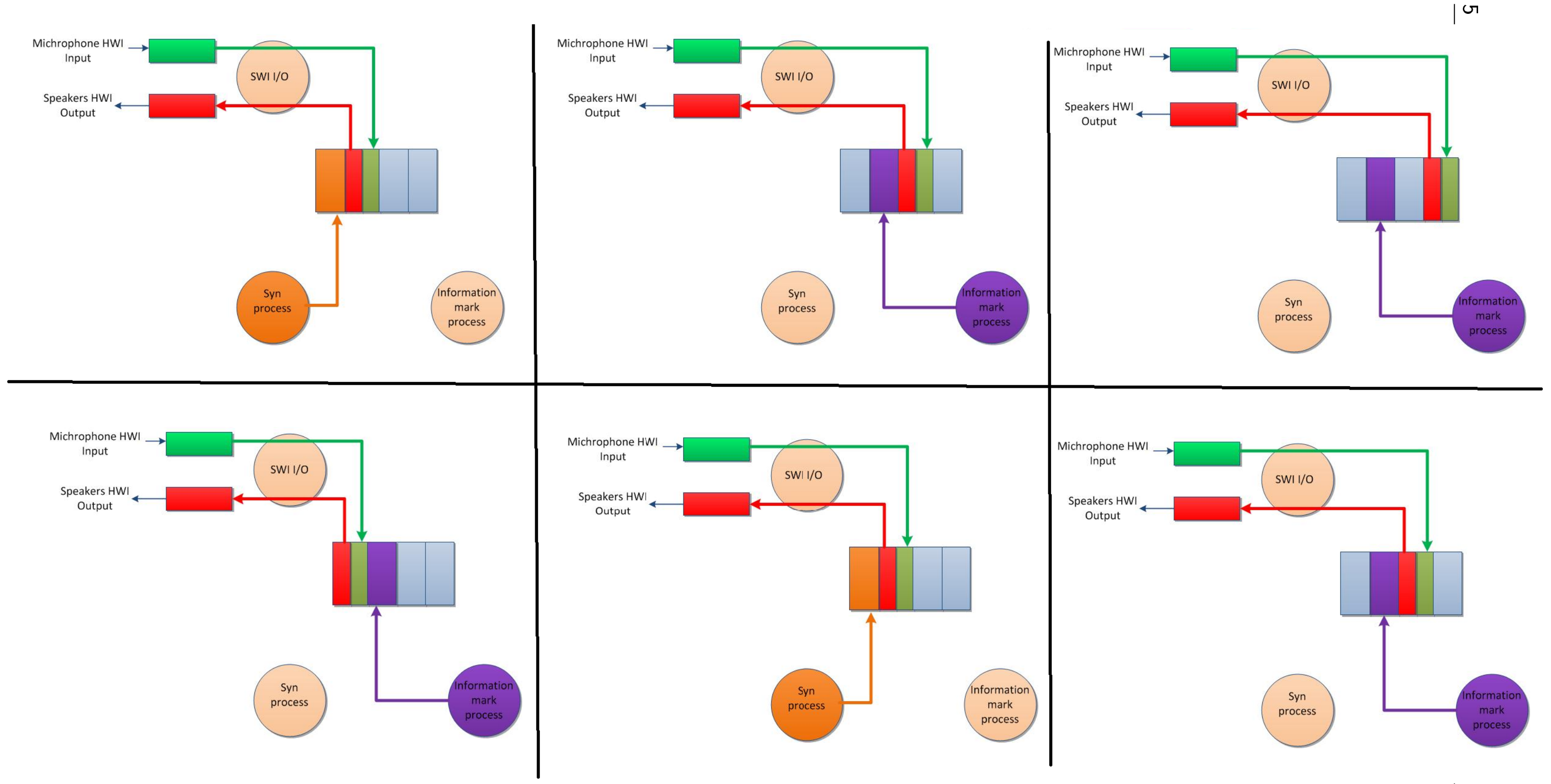

Fig. 4: Embedder software threads synchronization

data can be buffered at the same time that is sent back to the speakers out and after that can be processed. In addition, the internal buffering is also mandatory and there must be a task checking constantly for the synchronization mark.

The tasks for the extracting process are detailed below.

1. The main procedure, which executes the hardware and software initializations and prepares the hardware for the Input/Output.

2. A software interruption (SWI) that moves the input data to the output and loads this data into a circular buffer.

3. A task () that checks constantly for the synchronization mark and posts a semaphore when it is found in order to extract the watermark using the following task.

4. A task (), that extracts the watermark into a given frame placed in the position indicated by the above task. Note that this task will be waiting for a semaphore while no synchronization marks are found.